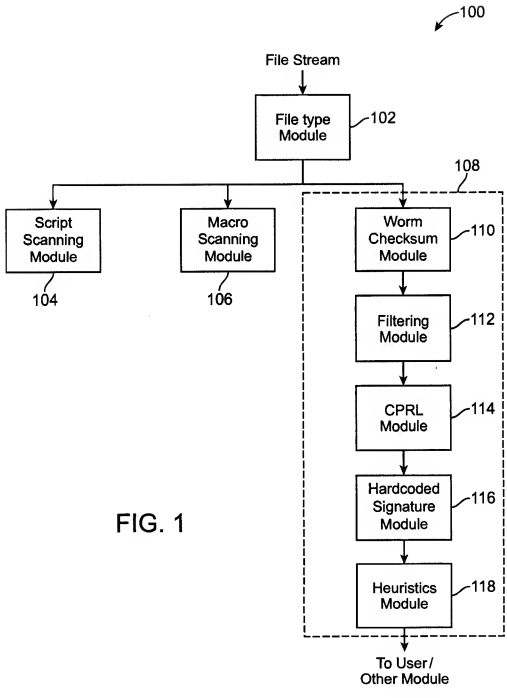
100
File Stream
File type Module
102
108
Script Scanning Module
104
Macro Scanning Module
106
Worm Checksum Module
110
Filtering Module
112
CPRL Module
114
Hardcoded Signature Module
116
Heuristics Module
118
To User / Other Module

FIG. 1

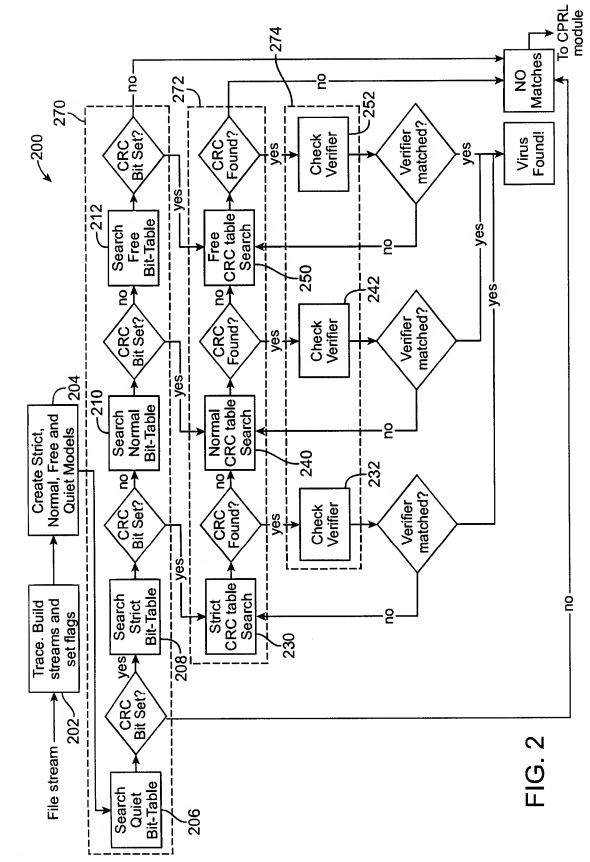
200
270
File stream
202
Trace. Build streams and set flags
Create Strict, Normal, Free and Quiet Models
204
Search Quiet Bit-Table
206
CRC Bit Set?
yes
Search Strict Bit-Table
208
Strict CRC table Search
230
CRC Found?
yes
Check Verifier
232
Verifier matched?
yes
no
CRC Bit Set?
no
Search Normal Bit-Table
210
Normal CRC table Search
240
CRC Found?
yes
Check Verifier
242
Verifier matched?
yes
no
CRC Bit Set?
no
Search Free Bit-Table
212
Free CRC table Search
250
CRC Found?
yes
Check Verifier
252
Verifier matched?
yes
no
272
274
Virus Found!
NO Matches
To CPRL module

FIG. 2

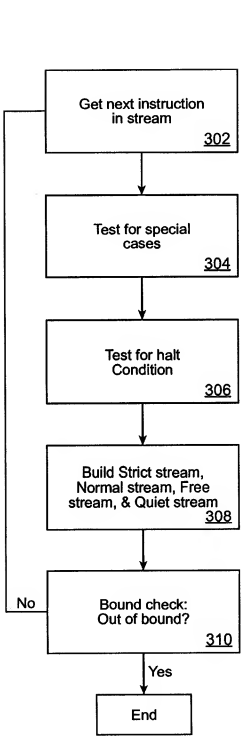
300
Get next instruction in stream
302
Test for special cases
304
Test for halt Condition
306
Build Strict stream, Normal stream, Free stream, & Quiet stream
308
Bound check: Out of bound?
310
No
Yes
End

FIG. 3

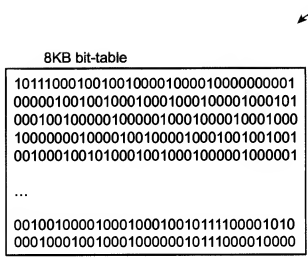
400
8KB bit-table
10111000100100000100001000000001
00000100100010001000100010001000101
00010010000010000010001000010001000
10000000100001001000010001001001001
00100010010100010010001000001000001
...
00100100001000100010010111100001010
00010001001000100000010111000010000

FIG. 4


500
502
504
CRC16: 2E
Index: 77
CRC16: FA
Index: 5
CRC16: FA
Index: 45
CRC16: 77
Index: 97
...
CRC16: 45
Index: 39
506

FIG. 5

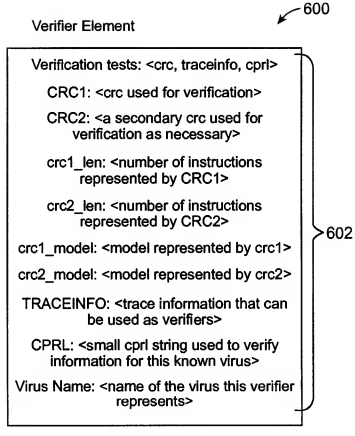
600
Verifier Element
Verification tests: <crc, traceinfo, cprl>
CRC1: <crc used for verification>
CRC2: <a secondary crc used for verification as necessary>
crc1_len: <number of instructions represented by CRC1>
crc2_len: <number of instructions represented by CRC2>
crc1_model: <model represented by crc1>
crc2_model: <model represented by crc2>
TRACEINFO: <trace information that can be used as verifiers>
CPRL: <small cprl string used to verify information for this known virus>
Virus Name: <name of the virus this verifier represents>
602

FIG. 6

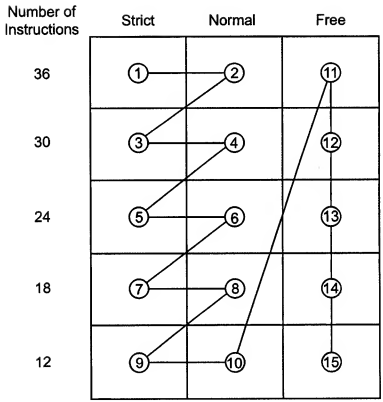
Number of Instructions
Strict
Normal
Free
36
30
24
18
12
1
2
11
3
4
12
5
6
13
7
8
14
9
10
15

FIG. 7

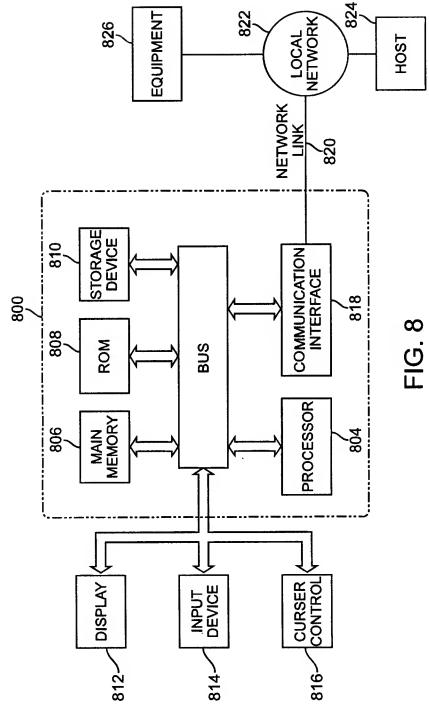
800
806
MAIN MEMORY
808
ROM
810
STORAGE DEVICE
BUS
804
PROCESSOR
818
COMMUNICATION INTERFACE
820
NETWORK LINK
822
LOCAL NETWORK
824
HOST
826
EQUIPMENT
812
DISPLAY
814
INPUT DEVICE
816
CURSER CONTROL

FIG. 8

Any two byte CRC contains a value from 0 to 65535. An eight kilobyte table contains 65536 bits. Therefore each bit in the 8k table can therefore represent a value from 0 (the first bit) to 65535 (the last bit). In other words, every value form 0 to 65531 can be represented by a single bit in and 8k table.

The following is an example of an algorithm that can be used to mask a CRC value to a single bit in the table:

C = CRC; B = Byte in table that contains the bit. b = the bit number that represents C. M = mask byte.

Note: Bit values 0 to 7 are represented by location, not by value.
Example:

1000000 = bit 0 set.
0000100 = bit 5 is set.

Algorithm

b = C ^ 8
B = C / 8

Set high bit only in M (1000000)
Shift byte right by count b. (if b = 0 then no shift occurs)
Bitwise AND M against B.
If NOT zero then match.

Pseudo Code

```
boolean
MASK (WORD C,           // CRC
      BYTE *TABLE)      // pointer to table
{
WORD  B;                // which byte in table
BYTE   b;               // bit count
BYTE   M;               // bit mask
BYTE   *N;              // pointer to byte in table

M = 0x8000;             // initialize to binary 10000000

B = C / 8;              // number of byte in table
b = C ^ 8;              // remainder of C / 8
N = TABLE + B;          // point to byte in table
M = M >> b;             // shift right 0 to 7 bits

if (M AND *N)           // AND Mask and Byte N in table
      return true;      // bit is set
return false;           // bit is not set.
}
```

FIG. 9